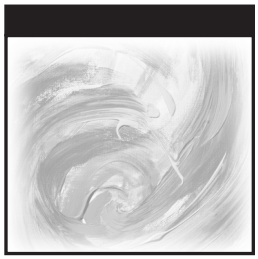
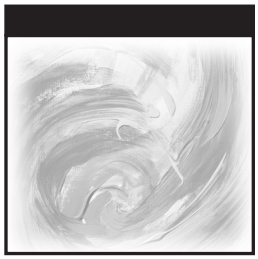


TCP/IP Analysis and Troubleshooting Toolkit

Kevin Burns

Wiley Publishing, Inc.



TCP/IP Analysis and Troubleshooting Toolkit

Kevin Burns

Wiley Publishing, Inc.

Executive Publisher: Robert Ipsen
Vice President and Publisher: Joe Wikert
Editor: Carol A. Long
Developmental Editor: Kevin Kent
Editorial Manager: Kathryn Malm
Production Editor: Pamela M. Hanley
Text Design & Composition: Wiley Composition Services

This book is printed on acid-free paper. ♾

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data: is available from the publisher

ISBN: 0-471-42975-9

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To my parents, who always believed in me

Contents

Acknowledgments

This book never would have been a reality without the following people: Emily Roche, who helped me open the door to writing and took me to my first book proposal seminar; Toni Lopopolo, who taught the seminar and put me in contact with my great agent Jawahara Saidullah. I want to thank Tony Fortunato for patiently reviewing my book for technical accuracy. Thanks also goes out to everyone at Wiley Publishing who worked so hard on this book, including my great development editor Kevin Kent, who held me to task on making sure readers would be able to easily understand the complex case studies and examples in the book. Last but not least, I want to thank my parents, who have given me everything and asked for nothing in return. This book is for you.

About the Author

Kevin Burns is the founder of Tracemasters, Inc., of Philadelphia, Pennsylvania, a consulting organization specializing in network analysis and training. Kevin's 10 years of experience consist of the design, implementation, and analysis of various multiprotocol, multivendor networks. This book comprises the techniques he has used in diagnosing complex network and application problems, which he also teaches to students at various seminars and corporate settings. Kevin can be reached at `kburns@tracemasters.com`.

Introduction

Why I Wrote This Book

Network engineers face difficult challenges on a daily basis. Servers can crash, WAN links can become saturated, and for unknown reasons, an application's performance can come to a crawl, pitting network engineers against application developers in a complicated blame game, usually without facts. Without the proper tools and training, when something breaks, network engineers often have to ask why: Why can't users obtain DHCP addresses, why can't users log into the server, and—the ever so bothersome question—why is the network slow? During all of this commotion, upper management is usually also asking why—Why haven't these problems been resolved? Most large network infrastructures have a mix of troubleshooting tools at their disposal, but more often than not the wrong tools are selected for the wrong job. How can you best use the tools at your disposal and the knowledge of your networks to assist you in quickly and decisively solving problems on your network infrastructures? The answer to that question is the subject of this book.

I wrote this book for the people on the front lines, the network field engineers. I have a great respect for field engineers. They are the doers, the people that make things work; they are also the first people whose pagers start beeping when things don't work. In my over 10 years of experience supporting desktops, servers, and large complex network infrastructures, I've come to the conclusion that the best field engineers are the ones who can solve the really tough problems.

People who are good problem solvers are usually tenacious and curious. These two qualities drive these people to stay up all night to try to solve a problem. They know the answer is there somewhere, waiting to be uncovered, and

they are tenacious enough to dig until they find it. The truly curious will most likely have read many good books on the TCP/IP protocol, including W. Richard Stevens' *TCP/IP Illustrated* (Addison-Wesley January 1994) and Douglas Comer's *Internetworking with TCP/IP (Prentice Hall January 2000)*. To date these books are the flagship manuscripts on understanding TCP/IP, but they focus intensely on theory and lack in practical examples. (That said, I still recommend every analyst have a copy of them on their bookshelves.) I have attempted to bridge the gap left by these two books by taking the most important concepts on the protocols and applying them to the most common problems a network analyst sees on TCP/IP networks. For the more curious, interested in the intricate details and inner workings of the protocol, I have provided an appendix further detailing the website.

The goal behind the *TCP/IP Analysis and Troubleshooting Toolkit* is to give the reader the information needed to successfully maintain the protocol in real-world networks. Since TCP/IP is the most common protocol in use today, this made the decision to concentrate an entire book on the subject of its analysis and troubleshooting methods easy. Rather than write a book about the many intricate and often-mundane details of the protocol, I attempt to empower you with the knowledge to understand and diagnose problems related to the TCP/IP protocol.

You will quickly notice that many of the examples in the book are either Cisco or Microsoft specific. Since those are the two most prevalent vendors in use today, I have chosen to use examples pertaining to their systems. The examples are by no means exclusive to either Cisco or Microsoft. In almost all cases, you can take the examples and apply them to any vendor's hardware or software. Specific examples that apply to a certain vendor are noted. Along this line, you might also notice several analysis tools mentioned or used in the examples. The type of tool is not typically important, just as long as it provides the functionality needed or described.

An understanding of the technology is what's important and that is what this book concentrates on.

Who Should Read This Book

Although this book does provide an introduction to network analysis techniques and the TCP/IP protocol, it is not for beginners. A basic understanding of the OSI model is important, as well as a decent level of experience managing server operating systems running TCP/IP.

More advanced readers already familiar with the protocol will benefit greatly from the case studies presented in each chapter. This book will help you become a better network analyst. If you are a network administrator eager

to learn more about understanding communications between clients and servers, this is a good place to start. If you are already familiar with configuring routers and switches, this book will teach you the technology behind the configuration commands; it will help you learn to think "outside the box."

This book is about technology and how to best use tools at your disposal to keep your networks running smoothly.

How This Book Is Organized

The book is organized into three parts:

- **Part I: Foundations of Network Analysis** answers such questions as "Why protocol analysis?" and "What tools do I use?" It explains the process of capturing and manipulating trace files. It also provides a refresher of the OSI model and the basic concepts of network communication that are needed to benefit from the material presented in the later chapters.
- **Part II: The Core Protocols** builds the foundation for understanding the protocols that TCP/IP is built upon. It is these protocols that provide the support for all other application-layer protocols.
- **Part III: Related TCP/IP Protocols** extends the search for understanding by revealing the inner workings of standard and vendor-independent protocol implementations. Applications such as DNS (Domain Name System), HTTP (Hypertext Transport Protocol), and FTP (File Transport Protocol) are thoroughly analyzed, and a deep investigation is conducted into Microsoft's TCP/IP implementation, including the ever-so-mysterious Server Message Block protocol.

In each chapter, the material is complemented with numerous case studies and examples from real, live networks. These examples and case studies are given to illustrate how the knowledge and techniques discussed can be put to use.

Tools

This book uses several different analysis tools to illustrate the troubleshooting examples. While the tools are not necessary to understand the examples, you do need them to view the trace files included on the companion Web site. The Web site includes instructions for downloading the freeware version of the Ethereal protocol analyzer, which can be used to view the traces.

The Companion Web Site

The companion Web site to this book (which can be found by pointing your browser to www.wiley.com/compbooks/burns) contains protocol standards such as RFCs (Requests for Comment), IETF (Internet Engineering Task Force) standards, and other resources concerning the protocols discussed in the book. It also contains online videos of most of the books example materials and trace files from the actually case studies, which you can load and examine for yourself. Finally, it includes several freeware and shareware utilities that are a must in the network analyst's toolkit. For more specific information as to what is on the Web site, see Appendix A.

PART

One

Foundations of Network Analysis

CHAPTER

1

Introduction to Protocol Analysis

What is protocol analysis? A *protocol* is defined as a standard procedure for regulating data transmission between computers. Protocol analysis is the process of examining those procedures. The way we go about this analysis is with special tools called *protocol analyzers*. Protocol analyzers decode the stream of bits flowing across a network and show you those bits in the structured format of the protocol. Using protocol analysis techniques to understand the procedures occurring on your network is the focus of this book. In my 10 years of analyzing and implementing networks, I have learned that in order to understand how a vendor's hardware platform, such as a router or switch, functions you need to understand how the protocols that the hardware implements operate. Routers, switches, hubs, gateways, and so on are simply nothing without the protocols. Protocols make networks happen. Routers and other devices implement those protocols. Understand the protocol, and you can largely understand what happens inside the box.

A Brief History of Network Communications

For years, complex processing needs have been the driving factors behind the development of computer systems. Early on, these needs were met by the development of supercomputers. Supercomputers were designed to service a single

application at a very high speed, thus saving valuable time in performing manual calculations.

Supercomputers, with their focus on servicing a single application, couldn't fully meet the business need for a computing system supporting multiple users. Applications designed for use by many people required multiple input/output systems for which supercomputers were not designed. These systems were known as time-sharing systems because each user was given a small slice of time from the overall processing system. The earliest of these systems were known as mainframes. Although not as fast as supercomputers, mainframes could service the business needs of many users running multiple applications simultaneously. This feature made them far more effective at servicing multiple business needs.

The advent of mainframes thus led to the birth of centralized computing. With its debute, centralized computing could provide all aspects of a networked communications system within a tightly controlled cohesive system. Such systems as IBM's S/390 provided the communication paths, applications, and storage systems within a large centralized processing system. Client workstations were nothing more than text screens that let users interact with the applications running on the centralized processing units.

Distributed computing followed on the heels of centralized computing. Distributed computing is characterized by the division of business processes on separate computer systems. In the late 80's and early 90's the dumb terminal screens used in centralized computing architectures started to be replaced by computer workstations that had their own processing power and memory and, more importantly, the ability to run applications separate from the mainframe. Early distributed systems were nothing more than extensions of a single-vendor solution (bought from a single vendor) over modem or dedicated leased lines. Because the vendor controlled all aspects of the system, it was easy for that vendor to develop the communication functions that were needed to make their centralized systems distributed. These types of systems are known as "closed" systems because they only interoperate with other systems from the same manufacturer. Apple Computer and Novell were among the first companies to deliver distributed (although still proprietary) networking systems.

Distributed processing was complicated. It required addressing, error control, and synchronized coordination between systems. Unfortunately, the communication architectures designed to meet those requirements were not compatible across vendors' boundaries. Many closed proprietary systems were developed, most notably IBM's System Network Architecture (SNA) and Digital Equipment Corporation's DECNet. Down the road, other companies such as Novell and Apple followed suit. In order to open up these "closed systems," a

framework was needed which would allow interoperability between various vendors' systems.

OSI to the Rescue

OSI (Open System Interconnection), developed by the International Organization for Standardization (ISO), was the solution designed to promote interoperability between vendors. It defines an architecture for communications that support distributed processing. The OSI model describes the functions that allow systems to communicate successfully over a network. Using what is called a layered approach, communications functions are broken down into seven distinct layers. The seven layers, beginning with the bottom layer of the OSI model, are as follows:

- Layer 1: Physical layer
- Layer 2: Data link layer
- Layer 3: Network layer
- Layer 4: Transport layer
- Layer 5: Session layer
- Layer 6: Presentation layer
- Layer 7: Application layer

Each layer provides a service to the layers above it, but also depends on services from the layers below it. The model also provides a layer of abstraction because upper layers do not need to know the details of how the lower layers operate; they simply must possess the ability to use the lower layers' services. The model was created so that in a perfect world any network layer protocol, such as IP (Internet Protocol), IPX (Internet Packet Exchange), or X.25, could operate regardless of the physical media it runs over. This concept applies to all of the layers, and in later chapters you can see how some application protocols function identically over different network protocols (and sometimes even different vendors—Server Message Block (SMB) is a perfect example of this as it is used by Microsoft, IBM, and Banyan's server operating systems). Most communication protocols map very nicely to the OSI model.

NOTE **OSI actually consists of not only the model but also a suite of complex protocols. Although the protocols are rarely used today, their original purpose was to provide a single protocol suite that all vendors could adopt into their systems, allowing for interoperability. The model survived, but unfortunately, the protocols did not.**

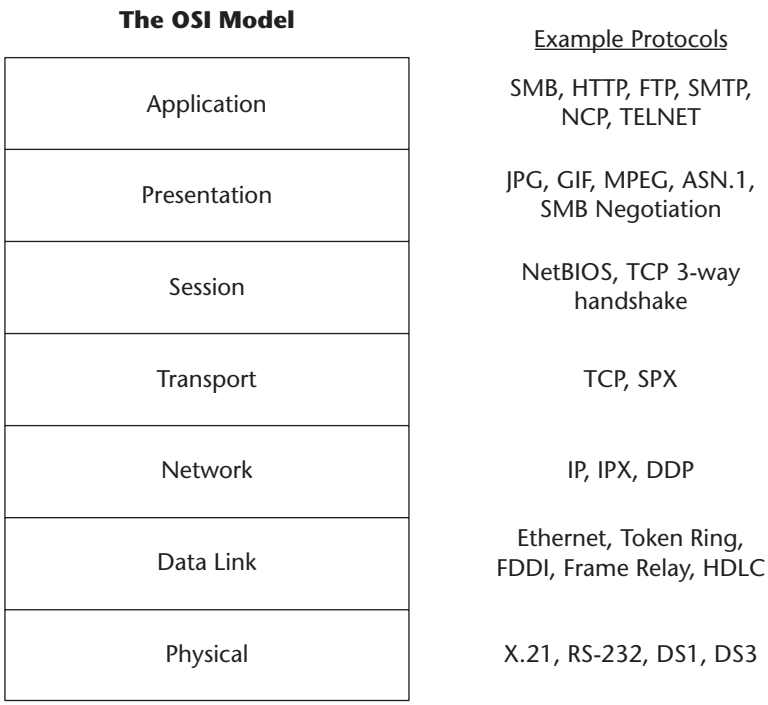


Figure 1-1 The OSI model.

Defining the Layers

Because almost all protocols are based on the OSI model, it is important to completely understand how the model operates, and to understand the protocols, you must first understand the framework. The following sections explain the seven layers in more detail, and Figure 1-1 gives examples of protocols that reside at each layer.

Layer 1: Physical Layer

The simplest definition of the physical layer is that it deals with how binary data is translated into signals and transmitted across the communications medium. (I talk more about media in the "Detailed Layer Analysis" section later in this chapter.) The physical layer also comprises the functions and procedures that are responsible for the transmission of bits. Examples would be procedures such as RS-232 handshaking or zero substitution functions on B8ZS T1 circuits. The physical layer concerns itself only with sending a stream of bits between two devices over a network.

Layer 2: Data Link Layer

Layer 2, the data link layer, handles the functions and procedures necessary for coordinating frames between devices. At the data link layer, zeros and ones are logically grouped into frames with a defined beginning and end. Unlike the physical layer, the data link layer contains a measure of intelligence. Ethernet, a common Layer 2 protocol, contains detection algorithms for controlling collision detection, corrupted frames, and address recognition. Higher layers depend on the data link layer not only to provide an error-free path but also to detect errors that may occur. Corrupted data should never be passed to upper layers.

Layer 3: Network Layer

Layer 3 is the end-to-end communications provider. Whereas the data link layer's responsibility ends at the next Layer 2 device, the network layer is responsible for routing data from the source to the destination over multiple Layer 2 paths. Applications utilizing a Layer 3 protocol do not need to know the details of the underlying Layer 2 network. Layer 3 networks, such as those using the Internet Protocol, will span many different Layer 2 technologies such as Ethernet, Token Ring, Frame Relay, and Asynchronous Transfer Mode (ATM). Some examples of Layer 3 protocols are IP, IPX, and AppleTalk Datagram Delivery Protocol (DDP). Although the network layer is responsible for the addressing and routing of data from source to destination, it is not responsible for guaranteeing its delivery.

Layer 4: Transport Layer

Networks are not reliable. On Ethernet networks, collisions can occur resulting in data loss, switches can drop packets due to congestion, and networks themselves can lose data due to overloaded links (the Internet itself experiences anomalies such as these on a daily basis). Protocols that operate in the transport layer may retransmit lost data, perform flow control between end systems, and many times add an extra layer of error protection to application data. While the network layer delivers data between two endpoints, the transport layer can guarantee that it gets to its destination.

Layer 5: Session Layer

The session layer provides the ability to further control communications between end systems by providing another layer of abstraction between transport protocols and the application. If an application layer protocol possesses this functionality, a session layer protocol may not be needed. NetBIOS, as you will see later in this chapter, is a perfect example of a session layer protocol. Sometimes the session layer does not reveal itself as a protocol, but rather as a

procedure performed to allow a protocol to continue its functions. Even though a protocol will exist at a certain layer, a procedure of that protocol can sometimes perform functions that normally reside in another layer. I will note instances in later chapters where this anomaly takes place.

Layer 6: Presentation Layer

The presentation layer is another layer that sometimes does not manifest itself in obvious ways. The presentation layer handles making sure that data formats used by application layer protocols are compatible between end systems. Some examples of Layer 6 would be ASCII, JPG, and ASN.1. Just as I indicated was the case with Layer 5, some protocol functions performed in other layers fit nicely into the description of the presentation layer.

Layer 7: Application Layer

Many people confuse Layer 7 with the applications used on servers or workstations. Application layer protocols are not user applications but instead the protocols that allow those applications to operate over a network. A user browsing the Internet with Internet Explorer utilizes an application layer protocol called HTTP. Microsoft Word users saving files to a network server make use of the Server Message Block (SMB) protocol. To a user, a network drive simply appears as G:\, but in the background there are powerful application layer protocols that allow G:\ to represent a location on a remote server. Other examples of application layer protocols are FTP and Telnet.

Protocol Analysis of the Layers

The following sections comprise a protocol analysis approach to the OSI model. They explain what each layer does and, more importantly, why. How each layer performs its function is left up to the protocol designers. I discuss how TCP/IP performs its functions in Chapters 3 through 6. More advanced readers may notice some vague or overly generic descriptions of packet descriptions in the following sections. I have written the descriptions this way to provide a generic blueprint for describing the layer's functionality; the details follow later in the book.

Layer 1: The Physical Layer

As I indicated earlier in the chapter, the physical layer concerns itself with how communications signals are transmitted across a medium. Appropriately, a medium is defined as a path where communication signals can be carried. A path is anything from copper, water, or air to even barbed wire if you can get the signals to successfully transmit over it. Media carry communication

signals. In wireless networks, signals travel over air as RF (radio frequency) radio waves. On 10BaseT Ethernet networks, they are carried as electrical voltage. In Fiber Distributed Data Interface (FDDI) networks, glass is used as the medium; the signals travel as pulses of light over glass fiber-optic cables. Many reasons exist as to why specific types of media are used in different technologies. Theoretically, you should be able to use whatever medium you want to carry the signals; unfortunately, the way those signals are represented places limitations on the types of media you can use.

Analog Signaling

Communications signals are transmitted in two ways. The first method, analog, is used to transmit signals that have values that vary over time. Sound is a perfect example of an analog signal. Sound is measured as an analog signal in cycles per second or hertz. The range of the human voice varies from about 100 Hz to 1,500 Hz. When early telephone networks were developed, it was difficult to create good-quality long-distance communications using analog signals because when these analog signals were amplified there was no way to distinguish the noise from the voice signal. As the analog voice signal was amplified, so was the noise. Converting analog voice signals to digital signals was one way to solve this problem.

Digital Signaling

Unlike analog signals, digital signals have only discrete values, either a one or a zero. Early digital telephone engineers figured out a way to modulate an analog signal onto a digital carrier using something called pulse code modulation, or PCM. PCM lets the instantaneous frequency of an analog signal be represented by a binary number. Instead of an amplifier having to guess at which signal to amplify, now it just had to repeat either a zero or a one. Using this method greatly improved the quality of long-distance communications. When computer data needed to be transmitted across network links, the decision to use digital signaling was easy. Since computers already represented data using zeros and ones, these zeros and ones could very easily be transmitted across networks digitally.

How these ones and zeros are represented is what digital signaling is all about. On 10BaseT Ethernet networks, data is represented by electrical voltage; a one is represented by a transition from –2.05 V to 0 V and a zero is represented by a transition from 0 V to –2.05 V. Over fiber-optic networks, a one might be represented by a pulse of light and a zero by the absence of light. The process isn't quite that simple, but the concept is basically the same. Different digital-signaling methods create ones and zeros on the media. Now, with the ability to have only two kinds of signals to recognize, it is much easier for amplifiers to pick out the digital ones and zeros from the background noise. With this ability to tell signals apart from noise, it became much easier to build networks capable of carrying computerized binary data over long distances.

NOTE **There are many types of digital signaling. One of the factors that drives the type of digital signaling used in a specific technology is its efficiency and method of bit representation. For example 10-Mb Ethernet uses what is called Manchester encoding (a type of digital signaling), but for 100-Mb Fast Ethernet, Manchester was inefficient if not impossible to use because the cabling available at the time (the late 1980s) couldn't support its high bandwidth. Instead, Fast Ethernet uses what is called Non Return to Zero Inverted (NRZI) encoding and in certain configurations Multi-Level Three (MLT-3). Other data link technologies use different digital signaling methods. Token Ring uses Differential Manchester and T1 circuits use AMI or B8ZS encoding.**

Layer 2: The Data Link Layer

So how do a bunch of ones and zeros become IP packets that traverse the network? For the network interface card (NIC) to put bits on the wire, it first must have a method of accessing the media. This method is called the media access method. All data link protocols designed for use in shared networks have one. One function of the media access method is letting the destination station recognize which bit is the first bit of the Media Access Control (MAC) frame. Once the first bit of the frame is found, the NIC can start grouping the ones and zeros into a Data Link Control (DLC) frame. Just as there are different methods of digital signaling, there are different types of DLC frames. In Ethernet, the IP protocol is carried by Ethernet II frames. On Token Ring, IP is carried by Token_Ring_SNAP frames.

NOTE **Since the objective of this book is to learn how best to analyze TCP/IP networks, I won't detail the many frame types that exist. For more information on the various frame types, refer to *Data Link Protocols* by Ulysses Black (Prentice Hall Professional 1993).**

It is important, however, to understand the basic details of Layer 2 framing. Each DLC frame has five basic parts:

- Media access portion
- Addressing
- Service access points
- Upper layer data
- Frame protection

These five basic parts are illustrated in Figure 1-2 and discussed in detail in the sections that follow.

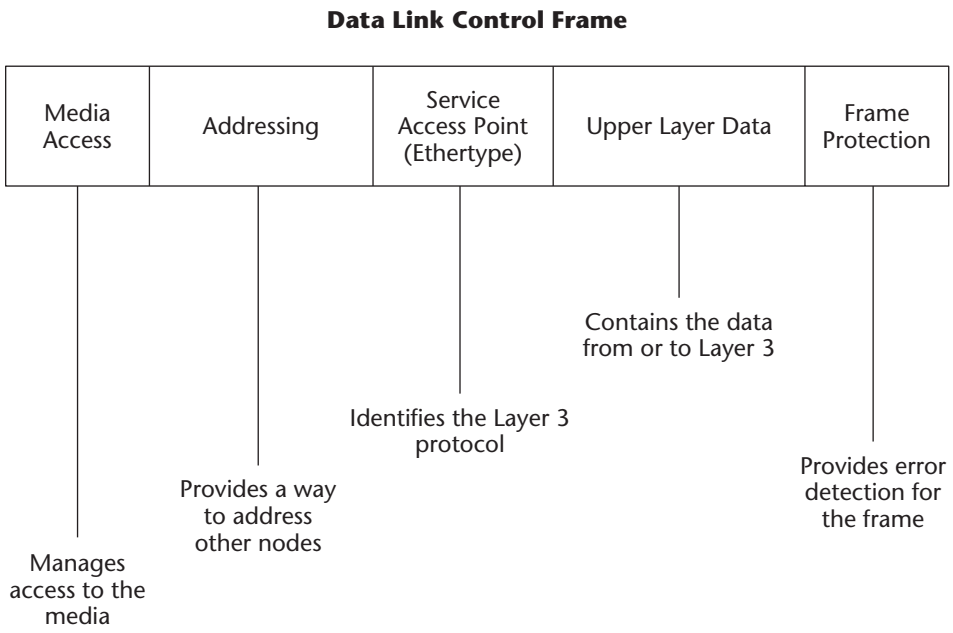


Figure 1-2 Data Link Control frame.

Media Access Portion

The media access portion of the frame consists of certain bit patterns and reserved bits for use by the NIC driver software. Media access means just what it says; the NIC must access the media. A NIC cannot always transmit at will; sometimes the media is being used by another node on the network. This scenario is where the term *shared networks* comes from. In a shared network only one node at a time can be transmitting bits out onto the wire. A shared network may physically consist of many wires and hubs, but logically it acts as one piece of wire. Only one station at a time may transmit on that wire. Consider the following examples:

- Ethernet uses a collision back-off algorithm called CMSA/CD. Using this algorithm, a station listens to see if the media is free and then transmits if it is. If it hears another station transmitting (this is called a collision) both stations back off for a certain time and try again until one station successfully obtains access to the media.
- Token Ring and FDDI use what is called a token-based access scheme whereby a small token frame circulates around the logical ring. When the token arrives at the appropriate station, that station marks the token as busy and attaches data to it for transmission around the ring.

Both methods have their benefits and drawbacks but the concept is essentially the same. Each data link protocol must provide some method for accessing the media.

MAC Addressing

Communication occurs between nodes on a network, and each node must have a unique identifier. This identifier is called the *Data Link Control address* or *DLC address*. It is also called the *MAC address*. (MAC is short for Media Access Control.) I use the two terms interchangeably throughout the book. MAC addresses are provided by the data link control endpoint, typically a NIC. (They are also known as *burned-in addresses* because the address is programmed permanently into ROM [read-only memory]. The process of creating a ROM chip actually involves *burning* small fuses inside of the chip to represent either a 1 or a 0, hence the name *burned-in-address*.) The MAC address is a 6-byte hexadecimal number that uniquely identifies an interface on a node. It is important to remember that the MAC address does not identify the node, but only an interface to it. Nodes can be workstations, servers, routers, bridges, or even access points into a wireless network, and any of these nodes can have multiple NIC cards (that is, endpoints) on the network. A router, for example, may have many interfaces. On the other hand, a server may have just two connections, one to the production LAN and one to a backup LAN.

There are three types of MAC addressing, and Table 1-1 illustrates the three types.

- **Unicast.** Processed by a single endpoint
- **Multicast.** Processed by multiple endpoints
- **Broadcast.** Processed by all endpoints

The first one, a unicast address contains 6 bytes (in hexidecimal) that make up the entire address. The second, a multicast address, also contains 6 bytes. The third address, the broadcast address, has the same 6 bytes but each byte is the same value "FF." Why is this?

Table 1-1 Three Types of MAC Addresses

TYPE	EXAMPLE
Unicast	00-00-0C-45-A9-D5
Multicast	01-23-7D-34-1E-9A
Broadcast	FF-FF-FF-FF-FF-FF

Half-duplex NIC cards, when not transmitting data, listen on the wire for a MAC frame containing their own address. For example, on Ethernet, a node hears another station's transmission and synchronizes on its bit pattern. When it recognizes the first bit of the frame, it looks at the first 48 bits to determine if the frame should be copied off the wire and sent to the upper layers. Why 48 bits? Because there are 8 bits in a byte; therefore, 48 bits equals exactly 6 bytes, the length of the MAC address.

NOTE **Ethernet is by its nature a half-duplex protocol. At the time of its creation, only shared hubs existed; there was no switching. When an Ethernet card is connected directly to a switch port, there are only two stations on the segment, the Ethernet card on the computer and the switch port. By turning off collision detection and allowing both the NIC and the switch to transmit at will, the connection becomes full-duplex. Full-duplex is really just the disabling of collision detection on both ends of a point-to-point Ethernet segment.**

Nodes on a network need to be able to transmit data frames to a single station, multiple select stations, or all stations. A frame transmitted to a single station is known as a unicast frame, one transmitted to multiple stations is a multicast frame, and one transmitted to all stations is a broadcast frame. When a NIC card sees its own address in the destination portion of a MAC frame, it copies the frame off of the wire and determines to what upper-layer protocol it should be passed. The multicast address operates the same way, except that nodes must be told to listen for a specific multicast address. Video multicasting applications use this technique to stream a single video stream to multiple clients on the same Layer 2 network. The broadcast address (FF-FF-FF-FF-FF-FF) is the one MAC address that all stations must listen to. When a station sees a destination address of all Fs, it must copy the frame from the wire and look at it, even if the data in the frame is destined for an upper-layer protocol that the station doesn't support. In that case, the NIC simply discards the frame.

MAC addresses also have a unique way of identifying the hardware to which they belong. The first 3 bytes of each MAC address are known as the *Organizationally Unique Identifier (OUI)*. Each vendor who manufacturers NIC cards requests an OUI from the Institute of Electrical and Electronics Engineers (IEEE). The vendor uses this 3-byte value as the first 3 bytes in the NIC cards it manufacturers and then assigns the remaining 3 bytes. Because the first 3 bytes are static and cannot change, the vendor can manufacturer around 1.5 million NIC cards using the remaining 3 bytes. Table 1-2 contains a list of common vendors' OUIs.

Table 1-2 Sample OUIs

OUI	COMPANY ASSIGNMENT
000102	3Com
00508B	Compaq
000142	Cisco
0002B3	Intel
0004AC	IBM
0020D8	Nortel
00007D	Sun Microsystems

Figure 1-3 illustrates a breakdown of different reserved bits in the MAC address format.

The first bit (broadcast bit) is always set to 1 in multicast and broadcast frames. The second bit (universal/local bit) is reserved for organizations that use nonpublic NIC cards. This bit lets organizations choose their own OUI without being concerned with which ones have already been reserved by other vendors. If the local bit is set to 1, you know you are seeing a NIC card that is not in public use. Non public NIC cards are used for specific purposes and are not often mixed with NIC cards publicly sold by NIC manufacturers.

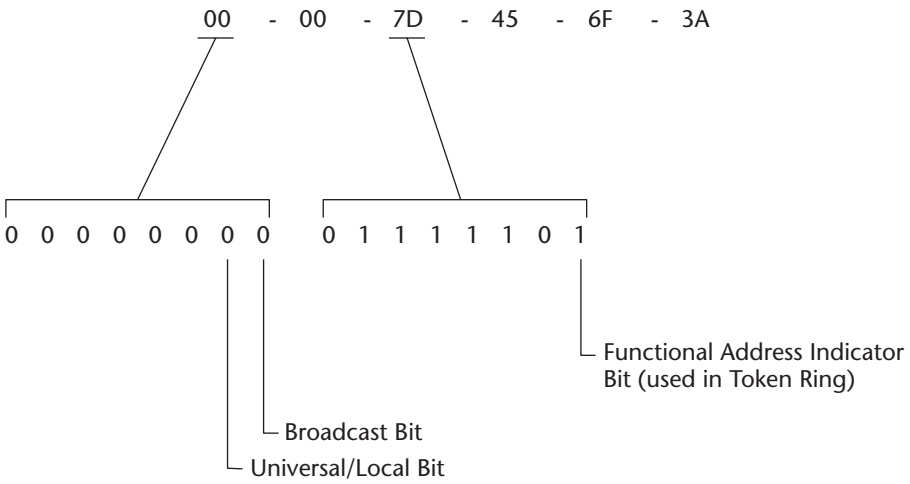


Figure 1-3 MAC address bit definitions.

MAINTAINING AN OUI LIST

Maintaining a handy reference of current OUI registrations can be very helpful in troubleshooting situations. Although most protocol analyzers have many OUIs already programmed into them, there are always new products on the market with OUIs your analyzer might not know about yet. When you are analyzing transactions at the data link layer, a handy OUI list makes it easier to spot which MAC addresses in the protocol trace belong to what hardware.

Several years ago I was analyzing broadcast traffic on a client's network and was seeing many strange IPX broadcasts on our IP-only segments. Comparing the MAC addresses from which the broadcasts were originating revealed that they all contained the same the 3 bytes (the OUI). After looking up the OUI, I realized that the broadcasts were coming from our print servers, which were incorrectly configured with IPX. Disabling IPX stopped the unnecessary broadcasts. The current list of OUIs registered with the IEEE can be found at `http://standards.ieee.org/regauth/oui/oui.txt`**.**

Ethertypes

At any given moment, a data link layer protocol is performing one of two tasks. It is either receiving a data link frame from the network and passing it to the network layer or it is receiving data from the network layer that needs to be transmitted out onto the network. The next couple paragraphs investigate this process further.

After the data link layer fully receives the frame, its next job is to determine the identity of the Layer 3 protocol to which the frame's data should be delivered. However, a workstation might be running multiple Layer 3 protocols besides IP; in mixed vendor environments, a workstation may have Novell IPX or AppleTalk running. How then does the data link layer determine which Layer 3 protocol should receive the data?

Inside the MAC frame there is a 2-byte field called an *Ethertype*. The value of this field determines what Layer 3 protocol should receive the data. Table 1-3 shows a partial listing of Ethertypes and their values.

Table 1-3 Sample Ethertypes

VALUE	DESCRIPTION
0000-05DC	IEEE 802.3 Length fields
0101-01FF	Experimental (For development)
0200	Xerox PUP—Conflicts with 802.3 Length field
0201	PUP Address Translation—Conflicts with 802.3
0600	Xerox XNS IDP

(continued)

Table 1-3 *(continued)*

VALUE	DESCRIPTION
0800	DOD IP
0806	ARP (For IP and CHAOS)
0BAD	Banyan Systems, Inc.
8137-8138	Novell IPX

In the reverse situation, when the data link layer is transmitting data passed down to it from a Layer 3 protocol, the data link layer simply places the correct Ethertype value inside the Ethertype field that corresponds to the Layer 3 protocol from which it received the data.

SERVICE ACCESS POINTS

Another method of upper-layer protocol identification is what are called service access points. Service access points are used with a frame type called the Logical Link Control, or LLC. LLC is more than just a frame format, it is an entire protocol used extensively in IBM Source Route bridge networks. Instead of Ethernet_II frames, LLC uses Ethernet_802.3 frames. It is also used by the NetBEUI protocol, which I will discuss in Chapter 3. Instead of an Ethertype, the LLC frame format uses a source service access point and a destination service access point, called SSAP and DSAP, respectively. The SSAP and DSAP values like Ethertypes tell the data link layer what upper-layer protocol should receive the data in the Layer 2 frame. Below is a decoding of an LLC frame. Instead of an Ethertype field, the 802.3 frame has a 2-byte length field. We can see that the SSAP and DSAP values are 0xF0, which tells the data link layer that the upper-layer protocol is NetBIOS.

```
802.3 Header
  Destination:          00:10:A4:AD:1E:75
  Source:               00:04:5A:76:F3:29
  LLC Length:           47
802.2 Logical Link Control (LLC) Header
  Dest. SAP:            0xF0  NetBEUI/NetBIOS
  Source SAP:           0xF0  NetBEUI/NetBIOS
  Command:              0x03  Unnumbered Information
NetBEUI/NetBIOS - Network Basic Input/Output System
  Length:               44
  NetBIOS Delimiter:    0xEFFF
  Command:              0x0E  Name Recognized(Wait)
  Option Data 1:        0x00  Reserved
  Session Number:       4
  Name Type:            0  Unique Name
  Xmit/Resp Correlator: 0x00000060
  Destination Name:     KEVIN_98       <0x00>
  Source Name:          SERVER         <0x20>
```

Upper-Layer Data

The purpose of MAC frames is to carry upper-layer data from one Layer 2 interface to another. Figure 1-4 shows two different examples of Layer 2 communications carrying upper-layer data.

NOTE **Data link protocols can carry multiple types of upper-layer protocols. Note that Figure 1-4 shows Ethernet encapsulating both IP and IPX.**

Frame Protection

After receiving all data from the network layer and before transmitting that data out to the network, the data link layer performs one more task to help protect the integrity of the data as it travels along the network path. At the end of each MAC frame it appends a 4-byte value called a CRC. This *CRC,* or *cyclical redundancy check,* is a value that is calculated by a complex formula based on the data inside the MAC frame. When the destination NIC receives the frame, it performs the same calculation on the data to see if the value is the same. If it is, the frame's data is passed on to the network layer. If not, the data link layer discards the frame since its integrity cannot be guaranteed. These types of errors, where a frame's contents are corrupted during the transmission over the local media, are called CRC errors. Figure 1-5 illustrates the CRC operation.

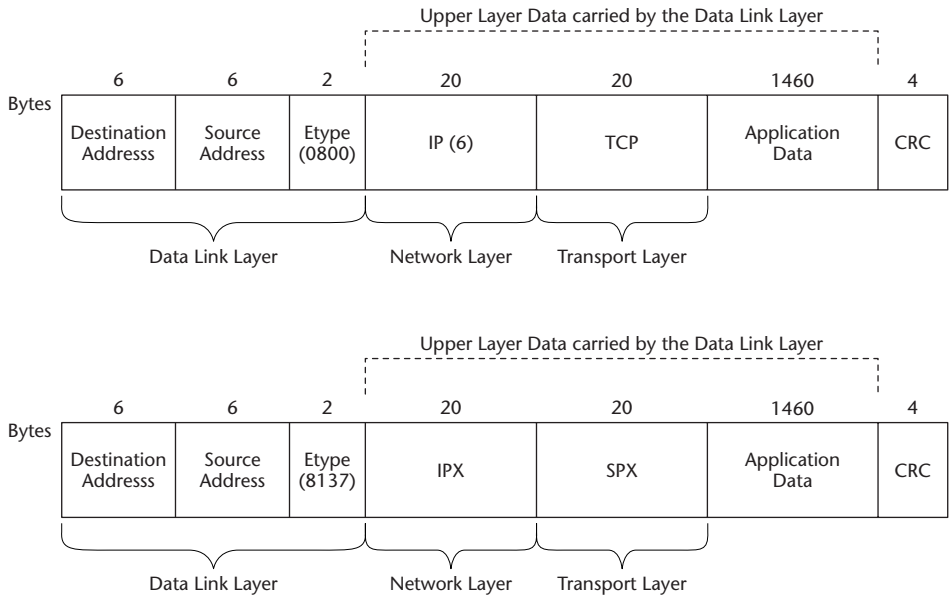
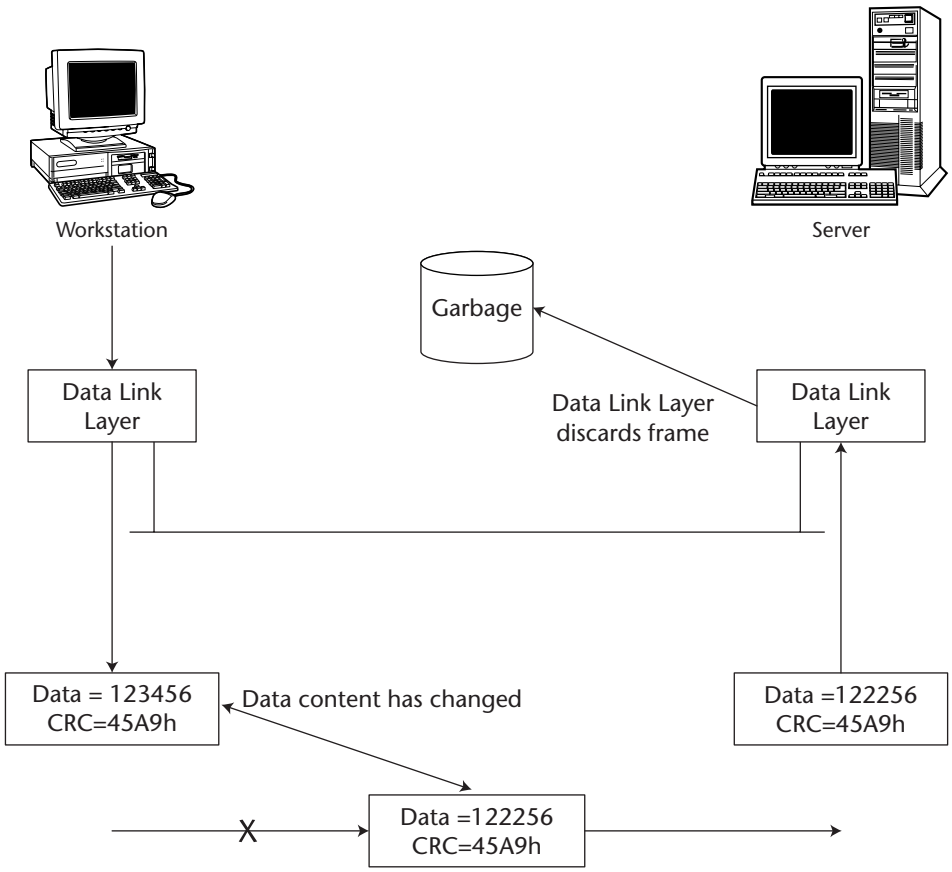


Figure 1-4 Layer 2 encapsulation examples.



1. The Data Link Layer calculates the 2-byte CRC value, appends it to the frame, and transmits the frame out on to the media.

2. As the frame is traveling over the media, its contents become corrupted.

3. The Data Link Layer on the destination receives the frame and performs the CRC calculation based on the frames contents. Because the resulting value is different, the NIC discards the frame.

Figure 1-5 CRC operation.

Layer 3: Network Layer

The network layer has four primary functions:

- Addressing
- Routing
- Path management
- Multiplexing

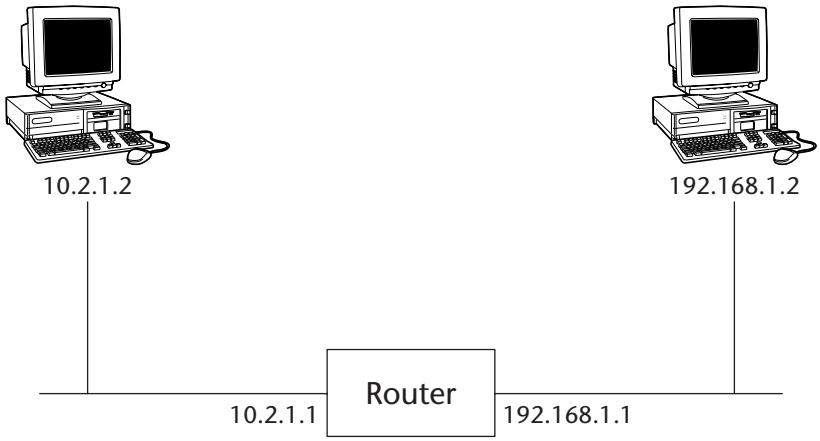


Figure 1-6 Routed IP network example.

The addressing function provides a Layer 2 independent address. Unlike a Layer 2 MAC address that can change as data is routed through an internetwork, the Layer 3 network address of an endpoint remains the same throughout the entire path.

Layer 3 addressing also provides the means for creating subnetworks for the purpose of logically partitioning a large Layer 2 LAN. Figure 1-6 illustrates two IP networks connected by a router. Notice how each subnetwork contains its own addressing scheme.

CROSS-REFERENCE **I discuss network addressing in more detail in Chapter 3.**

The network layer also provides for the end-to-end routing and delivery of datagrams through multiple networks. To accomplish this, protocols known as routing protocols distribute address reachability information throughout the entire network. Figure 1-7 shows a simple version of how the RIP (Routing Information Protocol) advertises information between routers. (Again, this material is discussed further in Chapter 3.)

The network layer must also handle path management issues such as rerouting around failed links, MTU (maximum transmission unit) discovery, and processing control information messages received from routers. ICMP (Internet Control Message Protocol) works in tandem with IP to provide critical information on the state of the network.

Also, because other upper-layer protocols use the services of the network layer, it must provide multiplexing and demultiplexing functions in order to pass data back and forth between layers. IP uses a very similar concept to Ethertypes, except that in the network layer they are called *protocol identifiers*. Table 1-4 shows a partial list of common protocols and their IP protocol IDs.

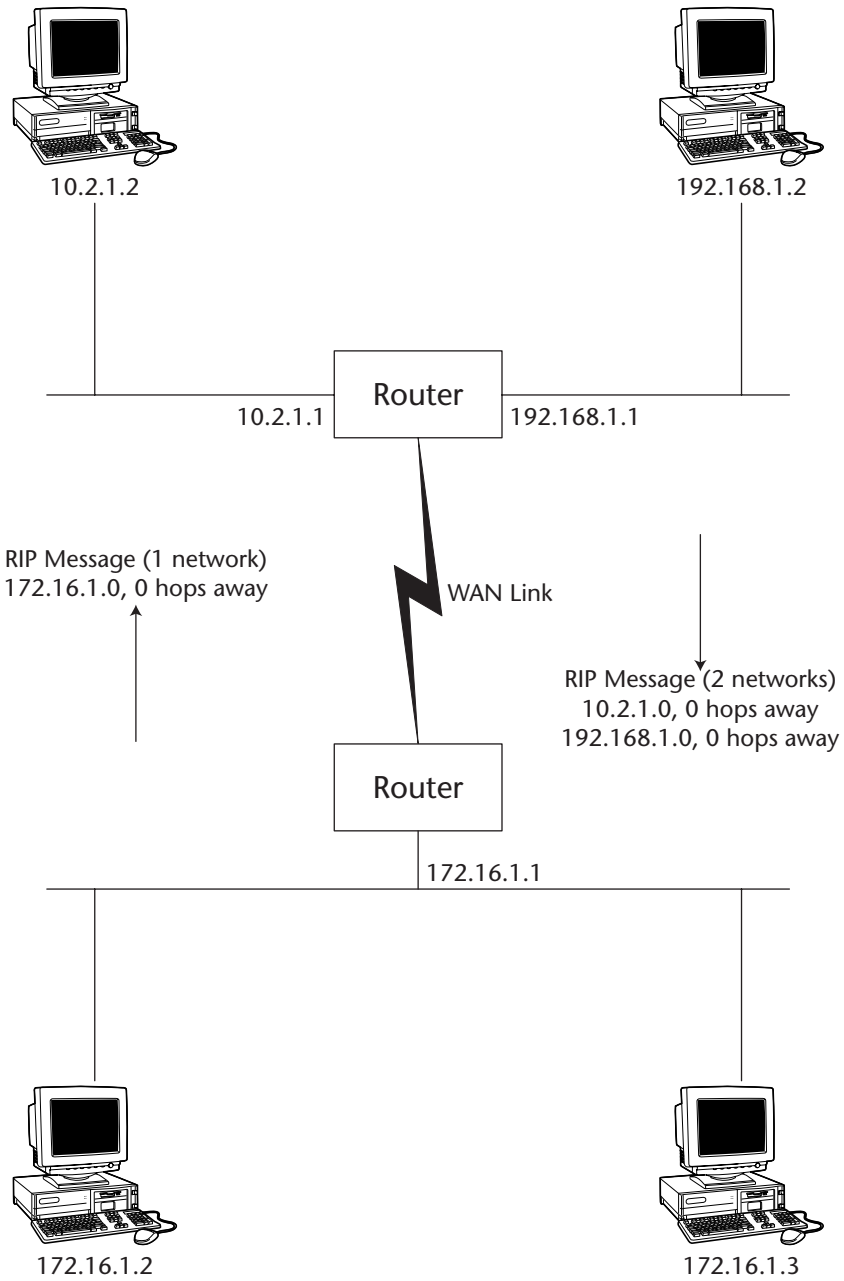


Figure 1-7 RIP operation.

Table 1-4 Example IP Protocol IDs

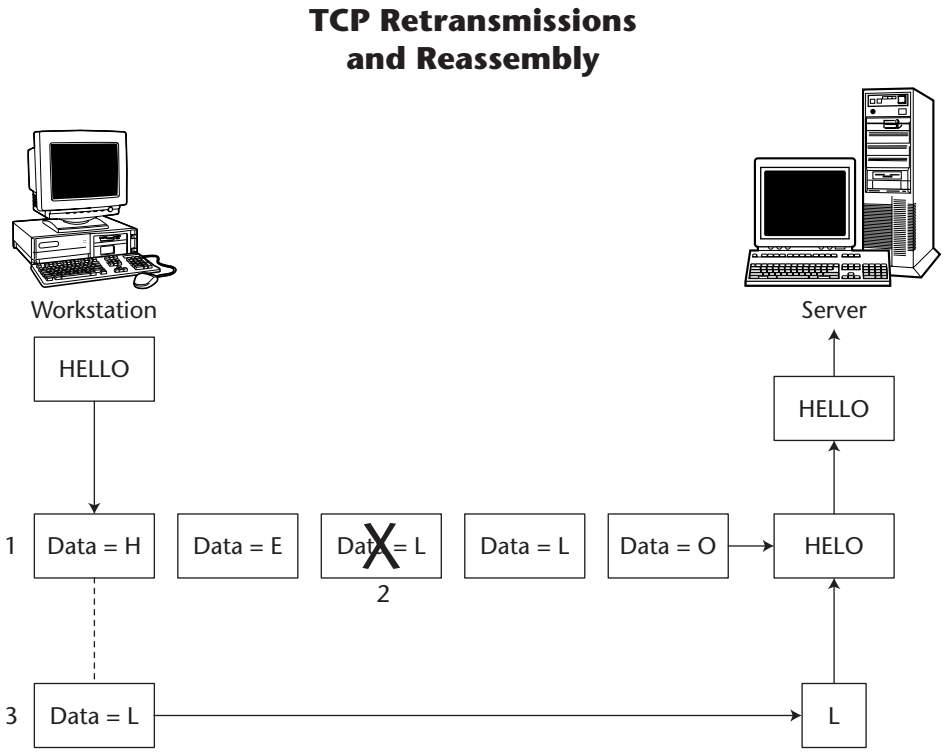
DECIMAL	KEYWORD	PROTOCOL
1	ICMP	Internet Control Message Protocol
2	IGMP	Internet Group Management Protocol
6	TCP	Transmission Control Protocol
17	UDP	User Datagram Protocol
37	DDP	Datagram Delivery Protocol
41	IPv6	Ipv6
50	ESP	Encapsulating Security Payload
51	AH	Authentication Header
83	VINES	Vines

Layer 4: Transport Layer

The transport layer can provide reliable or unreliable service. Why would any application developer want to use unreliable services when reliable services are available? The choice depends on the nature of the application. In the context of the transport layer, it makes sense to define what is meant by reliable and unreliable:

- Reliability in the transport layer refers to the ability of a transport protocol to provide some guarantee of the delivery of data over a network. By providing a guarantee, the data delivery becomes reliable.
- Unreliability in the transport layer refers to the lack of a transport protocol's ability to guarantee data delivery over a network.

As I stated earlier in the chapter, networks are unreliable. A number of events can occur in the lower three layers that may need to be handled by the transport layer. The transport layer needs to provide a method of detecting packet loss so that it can retransmit the lost data. Sometimes the network layer may route multiple packets over separate links, causing them to arrive at the destination in the wrong order. The transport layer must have a means of reassembling them into the correct order so that the data can be passed to the application. Since most applications exchange data in a structured format, the data needs to be reassembled into the proper order in which it was sent. Figure 1-8 illustrates an example of data being lost during its transmission over a network and the subsequent retransmission of the data by the transport layer protocol (in this case TCP).



1. Data handed down to the transport layer is broken up into multiple data segments and transmitted across the media.

2. One of the data frames is dropped by the network during transmission and the receiving station receives only four of the five segments.

3. After a time period, the transport layer retransmits the lost segment of data.

Figure 1-8 Reliable transport protocol example.

The transport layer must accommodate both of these situations. The answer to the question of why you might not want a reliable transport layer is that the choice of reliable versus unreliable services depends on the type of information being exchanged by the application. Obviously, a user saving a critical finance spreadsheet to a network server wants reliability in case a packet or two is lost during the file transfer. In that case, the transport layer simply retransmits the data and all is well, because this is how the transport layer provides reliability. However, consider the example of a phone call being routed through an IP network. Would it make sense to retransmit all data that might be lost during the conversation? Every time a packet containing voice was lost,

the transport layer would have to retransmit it behind voice data already received by the user. That retransmission would obviously lead to very garbled reception on the receiving end of the call. Could the transport layer wait and hold transmitted data in a buffer until the lost packets are retransmitted? It certainly could, but with the added delay for retransmission and reassembly, the quality of the voice call would be severely degraded. Thus, it would be preferable to use an unreliable protocol for transmitting voice data over an IP network.

CROSS-REFERENCE **In Chapters 5 and 6, I discuss two types of transport layer protocols, UDP (User Datagram Protocol) and TCP, and the purposes for their use.**

Layer 5: Session Layer

The session layer is the layer about which people most commonly ask, "Why do we need this layer? Don't other layers already possess the same functionality?" The answer is yes, other layers do possess this functionality, but the concept of a session layer still exists whether it exists in the form of a protocol or not. Further, many times an application layer protocol needs the services of the session layer to take on extra functions such as connection establishment and maintenance or data segmentation and reassembly. Figure 1-9, which appears later in the chapter, shows two types of session layer activities.

The first is a name resolution function that allows a host to determine the IP address of another host, given its name. The second, in this example, is the session layer setup performed by NetBIOS. It can only be performed once the IP address of the destination host is known. The degree that an application understands the heuristics of the transport layer determines its dependence on transport layer protocols. In pre-Windows 2000 environments, Microsoft's Server Message Block (SMB) protocol needed session layer services of NetBIOS since it could not natively communicate with transport layer protocols like TCP. In the discussion of the SMB protocol in Chapter 8, I show how SMB relies on the session layer to segment blocks of data for delivery to the transport layer.

Layer 6: Presentation Layer

The presentation layer is the most difficult to analyze simply because it sometimes does not exist in the sense of being a working protocol format. The presentation layer deals with how information is represented, how it is exchanged, and what structure it is stored in. The best example of the presentation layer is the method of data exchange between applications. For example, ASN.1 is a data format used in the SNMP protocol for querying the Management Information Databases (MIBS) on network devices. ASN.1 is the format

used to make the request. Application layer protocols utilize the services of the presentation layer, and in the case of ASN.1, SNMP utilizes its format (that is, its presentation). In the case of ASN.1, not only is it used as a representation of data, but it also specifies the methods for exchanging the data. For example, an SNMP MIB file will define the types of information a device supports. It will also specify the types of methods the device supports for obtaining that information. SNMP traps are a fine example of how the ASN.1 format is used in MIB files to define a method of information exchange. An SNMP trap is when, based on certain conditions, a device will send out (trap) information to a centralized management system.

Layer 7: Application Layer

The application layer handles the exchange of information. All software, such as word processors, browsers, and email clients, in some way exchange information. A user opening up a Web page is viewing information; a user saving a document is storing information. Application layer protocols contain the functionality to perform these tasks. Application layer protocols must handle situations that arise in data storage such as what happens when two users attempt to open a file simultaneously or how many attempts should be made at saving a file before notifying the user of an error. The application layer is the last line of responsibility in interpreting events in all lower layers.

Putting It All Together

Figure 1-9 shows (and this section explains) what goes on behind the scenes inside the OSI model when a user opens a file on a network server.

1. The open file request is passed from the client software to the application layer protocol, in this case, the Server Message Block (SMB) protocol. SMB cannot act on its own; it must pass the request down to the awaiting Session layer protocol, NetBIOS.
2. Upon receiving the SMB open file request, NetBIOS must perform two functions of its own. First, it must resolve the destination host name\\Server to an IP address. Second, it must open up a NetBIOS session with the destination host using this IP address.

NOTE **Take a look at the processes that have taken place so far—File Save Request, Name Resolution, Session Setup, and Transport Layer Connection. It's easy to see how many things have to happen even before the file can be transmitted across the network.**